THE CLUB
MeGAZINE
Ano VIII - Nº 95 Agosto 2002
DataSnap - Interfaces - Parte III
Desenvolvendo Aplicações para Celulares com Delphi
Criando Componentes Visuais - Parte 1
Acessando Oracle 9i via Internet através do Listener
Homologando seu Software para realizar Transferência Eletrônica de Fundos
Trabalhando com string no Delphi 6
Detectando URLs no RichEdit
Perguntas & Respostas
R$ 5,80

Revista The Club Megazine - 08/2002

A utilização, reprodução, apropriação, armazenamento em banco de dados, sob qualquer forma ou meio, de textos, fotos e outras criações intelectuais em cada publicação da revista “The Club” são terminantemente proibidos sem autorização escrita dos titulares dos direitos autorais.

Copyright© The Club® 2002

THE CLUB

THE CLUB
Rua Acre, 950 - Avaré - SP - CEP 18.700-260
Informações: (0xx14) 3732-3689
Suporte: (0xx14) 3733-1588
Fax: (0xx14) 3732-0987

Internet
http://www.theclub.com.br
Cadastro: cadastro@theclub.com.br
Suporte: suporte@theclub.com.br
Informações: info@theclub.com.br

Dúvidas
Correspondência ou fax com dúvidas devem ser enviados ao - THE CLUB, indicando "Suporte".

Opinião
Se você quer dar a sua opinião sobre o clube em geral, mande a sua correspondência para a seção "Tire sua dúvida".

Reprodução
A utilização, reprodução, apropriação, armazenamento em banco de dados, sob qualquer forma ou meio, de textos, fotos e outras criações intelectuais em cada publicação da Revista "The Club" são terminantemente proibidos sem autorização escrita dos titulares dos direitos autorais.

Copyright© The Club® 2002

Impressão e acabamento:
Impressos Gril - *Gril Gráfica e Repr. Ind. Ltda.*
Tel.: (0xx14) 3762.1345 - Fax: (0xx14) 3762.1259
Rua São Paulo, 447 - Cep 18.740-000
Taquarituba - SP
Tiragem: 5.000 exemplares

Diretor - Presidente
Celso Jefferson M. Paganelli

Diretor Técnico
Mauro Sant'Anna

Colaboradores
Claudenir C. Andrade - Mário Camilo Bohm
Marcio Alexandroni da Silva
Anderson H. Rodrigues

Delphi é marca registrada da Borland International, as demais marcas citadas são registradas pelos seus respectivos proprietários.

Editorial

Olá amigos.

Como todos já sabem, a cada nova versão o Delphi traz mais novidades. Veja nesta edição a nova unit StrUtils que traz novas rotinas para o trabalho com strings no Delphi 6, complementando àquelas já existentes no Delphi 5.
Você já deve ter notado que o Microsoft Word tem o reconhecimento de URLs. Muitos de vocês já desejaram que suas aplicações fizessem isto. Bem, nesta edição você verá uma matéria que realmente prenderá seus olhos. Nós vamos mostrar como criar esta característica no componente RichEdit.

E tem muito mais, leia e comprove.

Boa leitura a todos e até o próximo mês com mais novidades.

CJPaganelli

Celso Jefferson Paganelli
Presidente - The Club

Editorial 03
Desenvolvendo Aplicações para Celulares com Delphi 04
Criando Componentes Visuais - Parte 1 11
Acessando Oracle 9i via Internet através do Listener 14
DataSnap - Interfaces - Parte III 16
Homologando seu Software para realizar Transferência Eletrônica de Fundos 20
Trabalhando com string no Delphi 6 22
Detectando URLs no RichEdit 26
Perguntas & Respostas 29

Desenvolvendo Aplicações para Celulares com Delphi

Neste artigo veremos como desenvolver aplicações para celular através da tecnologia WAP.

O Delphi é muito versátil, e para quem já desenvolveu aplicações servidoras para Internet, verá a extrema facilidade em portar tais aplicações para celulares.

Mais adiante veremos como trabalha uma aplicação servidora, e agora daremos uma olhada no panorama atual do WAP no Brasil.

WAP no Brasil

Será que temos oportunidades de lucro no Brasil? A oportunidade existe, mas poucas empresas estão investindo nesta área, certamente lucrativa. Será que sou um romântico sonhador? Sonhando com o lucro em tempos de vacas magras? Amigos, posso afirmar que não, pois em 2000, quando lancei meu livro de WAP (Wap Guia de Tecnologia, Editora Brasport), o mercado estava muito aquecido, devido as fortes propagandas de empresas de telefonia em torno do WAP, mas na época, a barreira cultural de nosso país quase levou o sonho destas empresas por água abaixo. Sabem o que elas fizeram? Deixaram o WAP de lado e continuaram a investir em aparelhos móveis, mas destacando apenas sua beleza e algumas funcionalidades.

Facunte, então o sonho do lucro ficou ainda mais distante? Amigos, aí é que entra o consultor, para explicar aos gerentes de tecnologia, CEOs, que a relação custo benefício com o WAP é muito boa, e o desenvolvimento é quase transparente, pois podemos criar aplicações WAP na maioria das linguagens, como: Delphi, .NET (VB, C#, ASP), PHP, Java (JSP, EJB), entre outras.

Modelos de Aparelhos

A disponibilidade de aparelhos no Brasil é muito grande, e o preço é um forte atrativo, variando entre R$ 199,00 (modelo LG DM 160), até R$ 1.999,00 (Nokia 9210).

Imaginem uma equipe de vendas, ou até mesmo uma equipe de pesquisadores, com modelos mais baratos, em torno de R$ 199,00, fechando negócios, elaborando pesquisas, consultando clientes, disponibilidade de estoque entre outros.

Casos de Sucesso

Para provar a força do WAP, vamos conhecer alguns *casos de sucesso.*

Caixa Econômica Federal

A *Caixa Econômica Federal* disponibiliza a todos a informação sobre o saldo do FGTS através da tecnologia WAP. Para ter uma idéia da transparência neste *case*, a mesma aplicação disponibiliza informação via telefone(fixo), Internet e celular (WAP). Totalmente desenvolvida em JAVA, a solução teve seu custo reduzido devido a leve infra-estrutura necessária para abrigar

toda a aplicação.

Embora tenha exemplificado o *case* da *Caixa Econômica Federal*, é importante ressaltar que a maioria dos bancos nacionais oferecem variados recursos com a tecnologia WAP.

InvestShop.COM.BR

A instituição financeira InvestShop desenvolveu em conjunto com a EverSystem (talvez a maior empresa de desenvolvimento deste setor), uma solução bastante robusta, oferecendo ao usuário, um controle total de suas aplicações. Só para ter uma idéia, a aplicação disponibiliza até gráficos de ações em celulares WAP. É realmente incrível.

DETRAN-SP

Um dos pioneiros na tecnologia WAP, o DETRAN de São Paulo, disponibiliza informações de Multas, Pontuações na Carteira, entre outros.

Viram que o poder do WAP é grande, e basta uma idéia para transformar em lucro.

Algumas idéias para ganhar dinheiro

Bem, aqui vão algumas sugestões bastante interessantes para ganhar dinheiro com aplicações WAP.

Força de Vendas	Aplicação de auxílio a vendedores externos, com opção de consulta a estoque, faturas de clientes, e fechamento de pedidos.
Pesquisas de Campo	Aplicação que auxilia pesquisadores das mais variadas áreas: IBOPE, CENSO, Opinião, Produtos, entre outras
TimeSheet	Aplicação para TimeSheet de consultores, advogados, técnicos, entre outros
Processos	Gerenciamento de Processos Jurídicos, onde o advogado ou cliente poderá consultar informações sobre os andamentos dos processos.
m-Ticket	(m=mobile) Vendas de ingressos, passagens aéreas, cinemas, teatros, etc .
m-Finance	Aplicações para o mercado financeiro
m-Commerce	Comércio em geral através de aplicações WAP

Entendendo as aplicações WAP

Para entender um pouco como funciona uma aplicação WAP, precisamos conhecer o funcionamento de uma aplicação servidora.

Acredito que todos estão habituados no desenvolvimento de aplicações desktop, onde normalmente instalamos em cada terminal nossa aplicação. Isso é o que chamamos de aplicações locais.

Quem está acompanhando os excelentes artigos do meu amigo Anderson Hartel, teve um forte contato com aplicações distribuídas, onde são instalados módulos no servidor, e módulos no cliente (terminal).

Em nosso caso teremos apenas o módulo servidor, pois o cliente da nossa aplicação é um browser com suporte a tecnologia WAP, ou seja, a grande maioria dos celulares comercializados no mercado nacional.

Para facilitar a compreensão, a figura 1 ilustra bem o nosso caso:

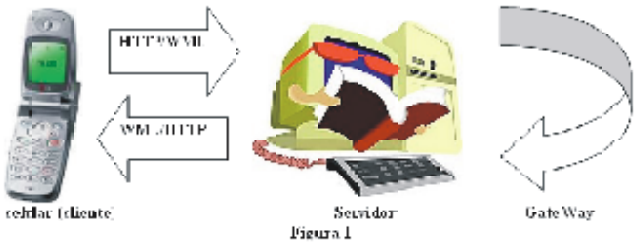


Figura 1

Na ilustração, temos um **celular**, que funciona como o nosso **cliente (terminal)**, onde solicita os serviços ao servidor. Os **dados (WML)** trafegam através do protocolo **HTTP**. Quando os dados chegam no servidor, o mesmo tem que interpretar a requisição, e esse serviço é feito pelo **GateWay.** O **GateWay** também tem como função empacotar o resultado da requisição, para que o servidor retransmita ao celular.

Em resumo necessitamos de duas aplicações básicas no servidor:

. Servidor HTTP (recomendo o Apache Server)
. Nossa aplicação servidora.

Um detalhe bastante interessante é que o **Apache Server** traz consigo embutido um excelente **GateWay WAP**. Embora esteja documentado que o **Apache** apenas suporta aplicações **WAP**, ele possui toda a implementação necessária para um perfeito funcionamento. Veremos adiante como conseguir o **Apache**, bem como a instalação e configuração num servidor.

Para que possamos prosseguir no desenvolvimento de nossa aplicação, precisamos também de um simulador **WAP**. Existem

vários no mercado, mas particularmente prefiro o **Deck-it da PyWeb.** A figura 2 ilustra a interface do **Deck-it.**

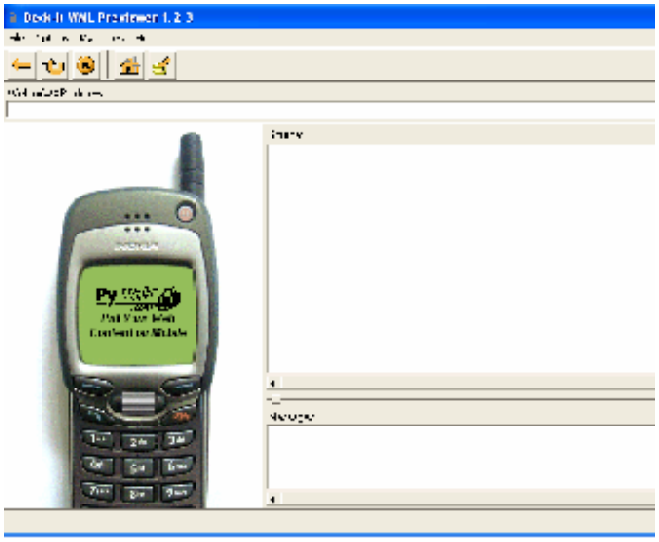


Figura 2 – Deck-it

Onde conseguir os softwares necessários

O **Apache Server** poderá ser "baixado" no site da **Apache.org (www.apache.org).** Procure "baixar" uma versão igual ou superior a 1.3.

O **Deck-it** está disponível no site da **PyWeb (www.pyweb.com),** e assim como o **Apache** é totalmente freeware.

Instalando e configurando o Apache.

A instalação do **Apache** no Windows é bastante simples e intuitiva. Já ouviram falar do famoso **next-next-next.** É basicamente isso. Após concordar com a licença do software, basta avançar o assistente de instalação, até a conclusão da mesma.

Agora vem a parte chata da coisa: configurar o bendito.

O **Apache** possui um arquivo com o nome **httpd.conf**, normalmente localizado no subdiretório **conf** da instalação. Exemplo **C:\Arquivos de Programas\Apache Group\Apache xx.xx\conf,** onde xx.xx é a versão do software.

Depois de localizado o arquivo **httpd.conf**, vamos editá-lo com o **Notepad** do Windows.

Apenas duas configurações são necessárias para que o **Apache** execute nossas aplicações WAP, *let´s go friends:*

Primeiramente localize dentro do arquivo, a seção DocumentRoot, a qual define o diretório base do nosso servidor HTTP. Especifique um diretório no seu computador, e faça a devida configuração nesta seção, como segue:

```
DocumentRoot "C:/DelphiWAP"
```

Neste caso, estou definindo como diretório base do meu servidor "C:/DelphiWap". Embora funcione de outra maneira, é interessante manter a barra deste jeito **/ ,** e não do modo tradicional \.

Para que o **Apache** permita a **execução** de nossas aplicações servidoras, é necessário dizer à ele em qual diretório podemos executar **scripts.**

Veja que isso é para uma maior segurança do servidor, pois nem todos os diretórios terão permissão para execução de **scripts**, ou aplicações servidoras. Localize a seção **ScriptAlias** e insira a configuração:

```
ScriptAlias /wap/ "C:/DelphiWAP"
```

Repare que estamos definindo o mesmo diretório da base, mas isto não é comum. Normalmente dividimos o diretório base em estruturas semelhantes a esta:

```
DelphiWap
   [scripts]
   [images]
   [data]
   [log]
   [html]
```

Em nosso caso não tem problema, pois não iremos disponibilizar o nosso servidor na Internet.

IMPORTANTE

Quem quiser disponibilizar o serviço na Internet, sugiro que faça uma estrutura de diretórios parecida com a minha sugestão, e defina o **ScriptAlias** apenas no subdiretório **[scripts].**

Outra coisa importante a saber, são os *MimeTypes WAP*. Mas o que são *MimeTypes?* Para que os servidores HTTP reconheçam um tipo de requisição, ou então, um método de envio, é necessário a definição dos *MimeTypes*.

Só para clarear um pouco a informação, uma imagem do tipo JPG, é definida como *image/JPG*.

Com isso o servidor saberá qual o melhor método de transmissão para este tipo de arquivo.

Mime Types do WAP

Tipo de Arquivo	Extensão	MIME Type
Código-fonte WML	.wml	text/vnd.wap.wml
WML compilado	.wmlc	application/vnd.wap.wmlc
Código-fonte WMLScript	.wmls	text/vnd.wap.wmlscript
WMLScript compilado	.wmlcs	application/vnd.wap.wml-scriptc
Imagem bitmap	.wbmp	image/vnd.wap.wbmp

Normalmente é necessário adicionar os *MimeTypes* em nossos servidores HTTP, mas em nosso caso, não há essa necessidade, visto que a partir da versão 1.3.1 o Apache já traz consigo toda a configuração especificada.

Colocando a mão na massa

Particularmente amigos, essa é a parte que mais gosto, mão na massa.

No Delphi iremos criar uma aplicação servidora no padrão CGI.

Peço licença aos saudosistas para utilizar o nosso famoso "Hello World" como exemplo.

A partir do Delphi, selecione as opções File/New/Other... e em seguida a opção Web Server Application, como ilustra a **figura 3**.

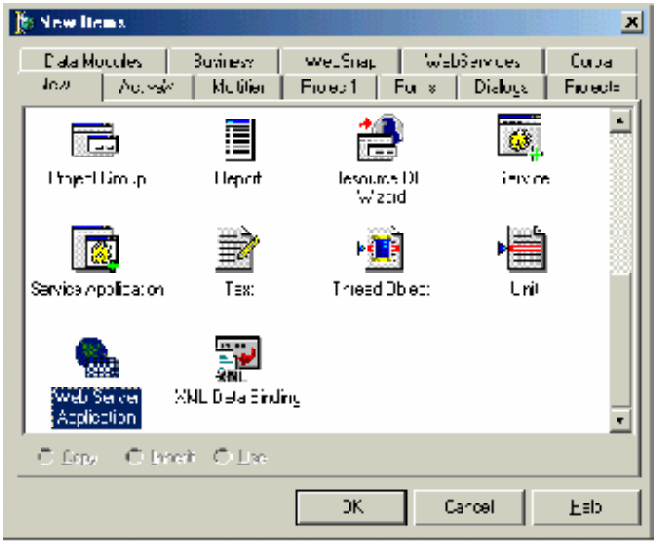


figura 3 opção Web Server Application

Na janela seguinte selecione a opção CGI Stand-Alone executable (***figura 4***).

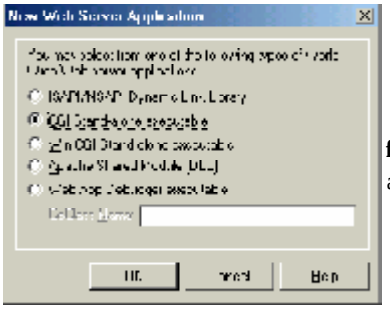


figura 4 seleção do tipo da aplicação

Em seguida teremos um ***WebModule (figura 5)***,

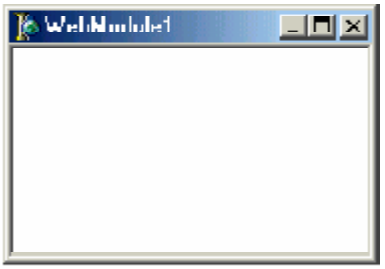


figura 5 WebModule

Mas o que é um *WebModule* ?

WebModule é um repositório de objetos, com a função de armazenar os objetos não visuais da aplicação tais como (TpageProducer, TqueryPageProducer, TQuery, etc), bem como responder as mensagens requisitadas pelo servidor HTTP.

Bem, para que uma aplicação servidora possa trabalhar, devemos delegar tarefas, através de *ActionsItems*, ou simplesmente *Itens de Ação*.

Para explicar melhor o uso de *ActionsItems*, imagine uma aplicação para inclusão e alteração de clientes. Teremos o seguinte cenário:

```
Aplicação (clientes.exe)
  ::: ActionItem(inclusão) – ação para incluir cliente
  ::: ActionItem(alteração)- ação para alterar cliente
```

É bastante simples, cada *ActionItem* tem uma função específica dentro da aplicação servidora. Para executar no browser ou celular uma determinada *ActionItem*, basta fazer como no exemplo:

http://site/scripts/clientes.exe/inclusao

Repare que informamos o nome da aplicação (clientes.exe) e o

nome da *ActionItem (/inclusão)*. A aplicação servidora não possui limite de *ActionItems*, portanto podemos criar aplicações complexas.

Bem, seguindo o nosso primero projeto, através do duplo-clique no WebModule, acesse o editor de *ActionItems* (***figura 6)***.

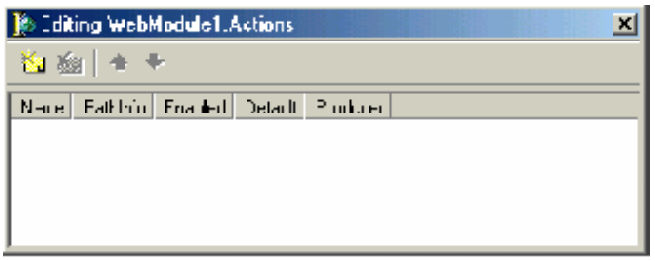


figura 6 editor ActionItems

Clique no primeiro botão do editor para inserir uma nova *Action* (***figura 7)***.

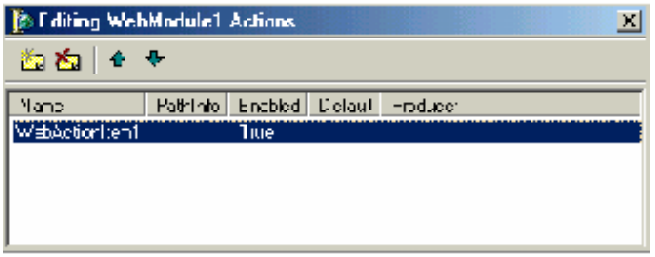


figura 7 ActionItem

Em seguida altere as seguintes propriedades**.**

OBJETO		
	TWebActionItem	
Objeto	Propriecade	Valor
padrão	Default	True
	Name	padrão
	PathInfo	/padrão

Esta será nossa *Action padrão,* ou seja, caso o usuário não digite nada, além do nome da nossa aplicação, esta *Action* será executada.

Embora a propriedade *PathInfo* possua o mesmo valor da propriedade *Name*, é ela que executa a *Action,* ou seja, no browser o que vale é o valor da *PathInfo.*

evento *OnAction* coloque o seguinte código:

```
001 Response.ContentType:='text/vnd.wap.wml';
002 Response.Content:='<?xml version="1.0"?>'+
003 '<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1
    //EM""http://www.wapforum.org/DTD/wml_1.1.xml">'+
004 '<wml><card id="cartaum"><p>Ola Mundo!</p>
    </card></wml>';
```

Não se assustem com as denominações do cabeçalho "<"DOCTYPE wml....", isso é um padrão, e o restante iremos conhecendo aos poucos.

Vamos analisar o código:

Na linha 001, estou dizendo ao servidor, através do método **Response.ContentType**, que o tipo de informação é no padrão WAP.

```
Response.ContentType:='text/vnd.wap.wml';
```

Em, seguida, nas linhas 002,003 e 004, estou empacotando a resposta (**Response.Content)** e enviando ao servidor, que por sua vez, envia ao celular.

```
Response.Content:='<?xml version="1.0"?>'+
   '<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//
EM"   "http://www.wapforum.org/DTD/wml_1.1.xml">'+
'<wml><card id="cartaum"><p>Ola Mundo!</p></card></
wml>';
```

Grave a *unit* do nosso projeto como *un_ola.pas* e o projeto como *ola.dpr*.

Normalmente em aplicações desktop, executamos através da tecla F9 (Run), neste caso, apenas iremos compilar a aplicação, e executá-la através do nosso simulador **Desk-It.**

No **Deck-It** digite o que segue na barra de endereços:

http://localhost/ola.exe

A **figura 8** ilustra o resultado da nossa aplicação:

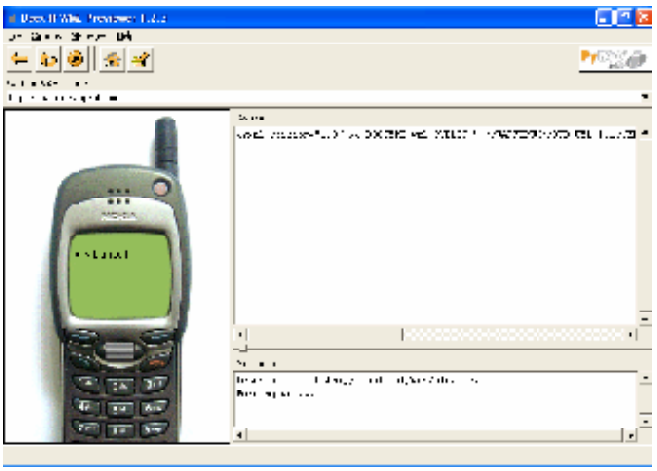


Figura 8 – Resultado da aplicação

Só para ter uma idéia, você poderia simular em seu próprio celular WAP este exemplo. Mas Facunte, como eu faria isso?

Conecte seu computador a Internet, anote o número do IP fornecido no momento da conexão, exemplo: 200.198.12.1, e digite no seu celular (o modo de entrada para comunicação WAP varia de celular para celular, consulte o seu manual de instruções), o endereço:

http://200.198.12.1/ola.exe

Repare que o número, na realidade é o seu endereço de IP atual (exemplo).

Continuando com a nossa aplicação, vamos criar mais uma *Action* com as seguintes propriedades:

OBJETO		
	TWebActionItem	
Objeto	Propriedade	Valor
Theclub	Default	False
	Name	theclub
	PathInfo	/theclub

No evento *OnAction* coloque o seguinte código:

```
001  Response.ContentType:='text/vnd.wap.wml';
002  Response.Content:='<?xml version="1.0"?>'+
003  '<!DOCTYPE wml PUBLIC"-//WAPFORUM//DTD WML 1.1//
     EM""http://www.wapforum.org/DTD/wml_1.1.xml">'+
004  '<wml><card id="cartaum"><p>Sejam bem-vindos ao
     The Club</p></card></wml>';
```

O código é exatamente igual ao anterior, estamos apenas substituindo a mensagem.

Vamos testar o código?

No **Deck-It** digite o que segue na barra de endereços:

http://localhost/ola.exe/theclub

Repare que estamos colocando o *pathinfo* ***theclub***.

A **figura 9** ilustra o resultado da nossa segunda *Action*.

Figura 9 – Segunda Action

É interessante, não acham?

Amigos, com isso concluímos o nosso artigo sobre o desenvolvimento de aplicações para celulares com Delphi.

Para testar o sucesso do artigo, gostaria de receber sugestões e críticas (emerson@facunte.com.br) e quem sabe, escrever uma continuação, com acesso a banco de dados, e tudo mais.

Como complemento de estudo a esse artigo, recomendo a visita nos seguintes sites:

www.aprendawap.com.br
www.pyweb.com
www.br.wmlclub.com
www.apache.org
www.linhadecodigo.com.br/wap/

Forte abraço e muito sucesso a todos.

Emerson Facunte

Emerson é Consultor de Tecnologia com diversos livros publicados, especialista em desenvolvimento de aplicações e-Business utilizando a ferramenta Delphi, baseado em WebSnap, DataSnap, BizSnap e ISAPI/Apache Modules.

Site: www.facunte.com.br

Criando Componentes Visuais - parte 1

Por Mauro Sant'Anna (santanna@mas.com.br). Mauro é um "MSDN Regional Director", consultor e instrutor da MAS Informática (www.mas.com.br), tendo ministrado treinamentos na arquitetura .NET desde outubro de 2000.

Esta é a primeira parte de um par de artigos que abordarão a criação de componentes visuais baseados em outros componentes já existentes, para serem usados em aplicativos .NET do tipo "WinForms".

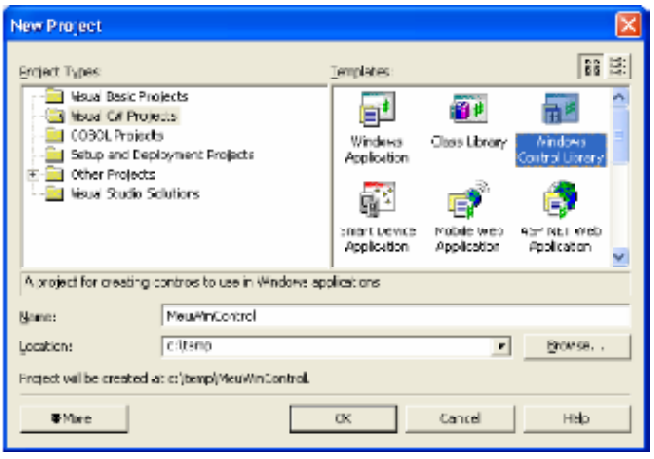
Tipos de componentes

Existem basicamente duas maneiras de criar componentes visuais para serem usados em programas WinForms no .NET Framework:

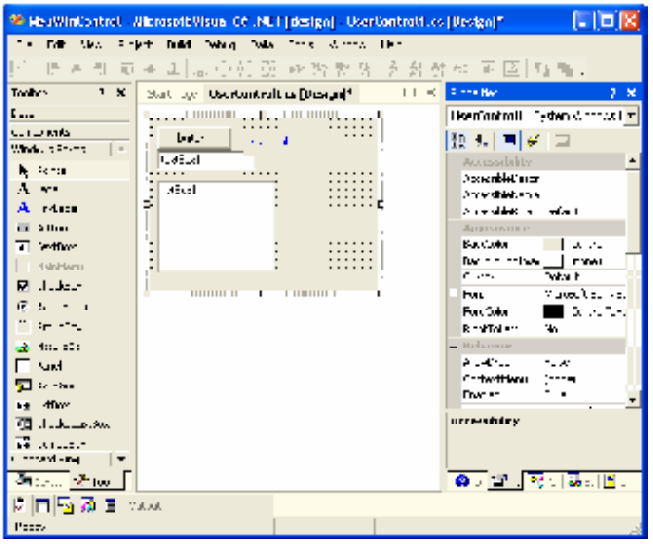
- WinControls;
- Classes derivadas de algum componente já existente.

WinControls

Os WinControls são componentes visuais que podem ser desenvolvidos de uma forma RAD, correspondentes a um projeto do tipo "Windows Control Library:



Eles são particularmente fáceis de criar por existir um bom suporte em tempo de desenvolvimento. É fácil colocar outros componentes sobre eles e também adicionar código para interceptar eventos dos componentes. Veja um WinControl sendo criado no Visual Studio .NET:



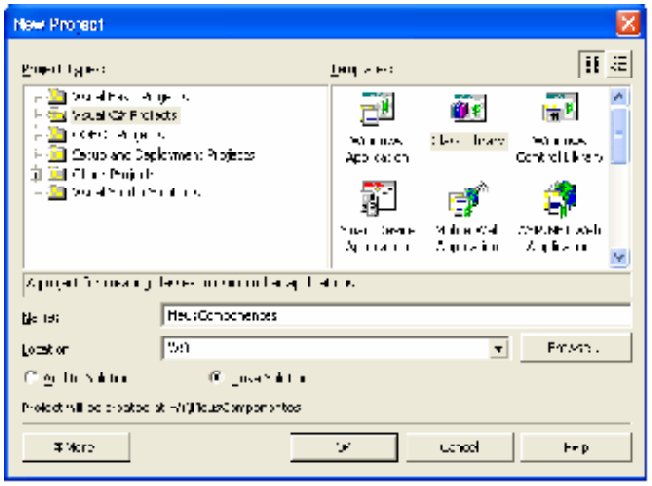
Os WinControls são bons em situações onde desejamos agrupar um conjunto de componentes em um "pacote" pronto para uso. No entanto, esta não é a ênfase deste artigo.

Derivando de um componente existente

Outra opção é mudar ligeiramente o comportamento de um componente já existente – qualquer componente. Esta técnica é particularmente útil quando você repetidamente altera um conjunto de propriedades de um componente ou intercepta sempre alguns eventos da mesma maneira.

Neste caso você quer que o componente continue funcionando basicamente da mesma forma, mas com um "algo mais". Veremos como criar um botão que se desabilita enquanto o evento Click *estiver sendo processado e também um* TextBox *que valida se o seu conteúdo é uma data.*

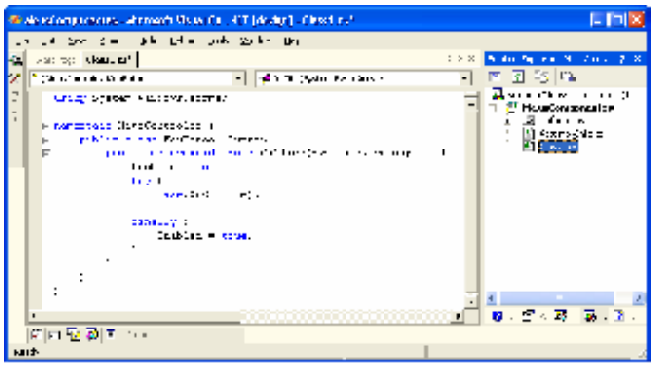
Inicialmente, crie um projeto do tipo "Class Library":



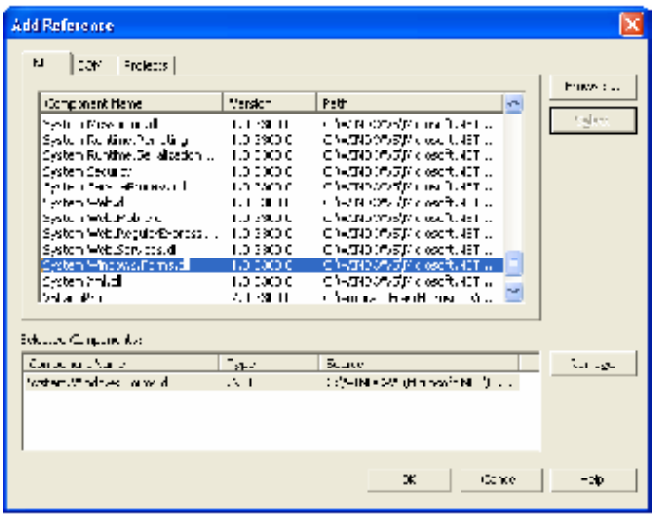
Entre o código do componente substituindo o que foi criado pelo assistente, por exemplo:

```
using System.Windows.Forms;
namespace MeusControles {
 public class MeuBotao: Button {
  protected override void OnClick(System.EventArgs e)
{
    Enabled = false;
    try {
     base.OnClick(e);
    }
    finally {
     Enabled = true;
    }
  }
 }
}
```

O código acima substitui completamente aquele criado pelo assistente:



Você deve adicionar uma referência à "System.WindowsForms.dll", pois é onde os componentes que usaremos como base estão localizados. Selecione "Project | Add Reference":



Depois de compilar o projeto, você pode testar o programa criando o componente em tempo de execução. Adicione um projeto do tipo "Windows Application" e adicione uma referência à DLL recém criada.

O código para instanciar o botão em tempo de execução é o seguinte:

```
// Método que será chamado quando o botão for
// pressionado
private void Clicou(object sender, System.EventArgs
     e) {
// Espera 3 segundos
 System.Threading.Thread.Sleep(3000);
}
// Cria meu botão
private void button1_Click(object sender,
     System.EventArgs e) {
 MeusControles.MeuBotao Btn = new
     MeusControles.MeuBotao();
 Btn.SetBounds(10, 50, 100, 20);
 Controls.Add(Btn);
 Btn.Text = "Aperte";
 Btn.Click += new System.EventHandler(Clicou);
}
```

Observe que o botão permanece desabilitado enquanto o método associado ao evento *Click* estiver sendo executado.

Interceptando métodos virtuais

Na maioria dos casos, incluindo o exemplo acima, desejamos interceptar algum evento no novo componente. Quando criamos uma classe derivada, no entanto, devemos interceptar um método virtual associado ao evento.

Todos os eventos da .NET Framework têm métodos virtuais

"associados". Isto não é obrigatório do ponto de vista do compilador, mas é uma convenção que a Framework segue e seus componentes devem seguir. Para descobrir o método virtual dado um evento, faça o seguinte:

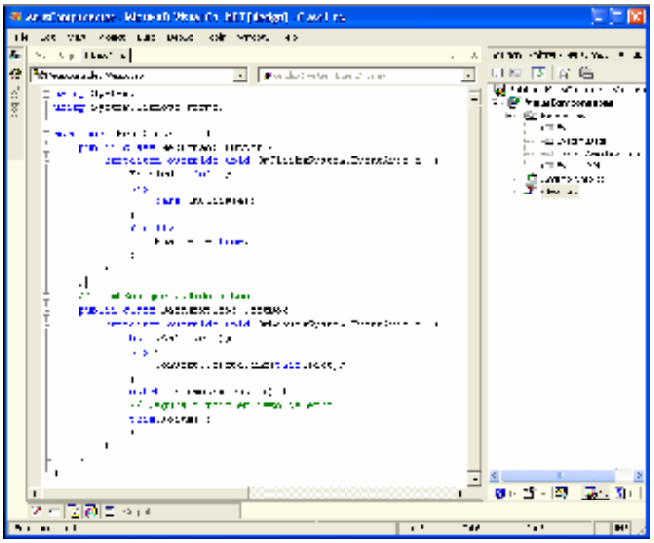
1. Acrescente *On* ao nome do evento
2. Para saber os argumentos, retire o "*ByVal Sender as Object*" e mantenha a estrutura seguinte.

Note no exemplo acima que estamos interceptando o método *OnClick*, associado ao evento *Click*.

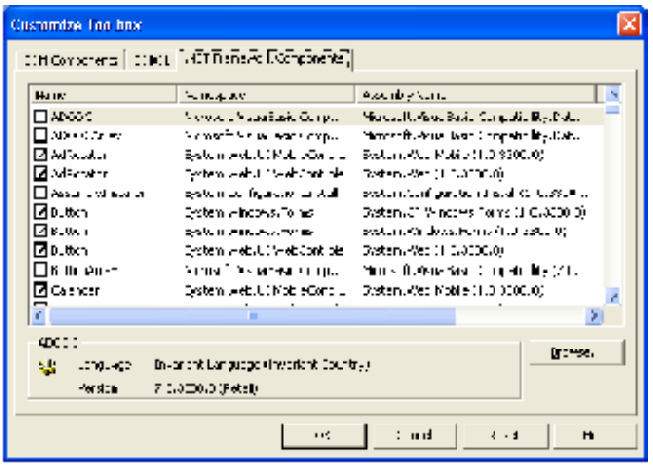
Veja outro exemplo de *TextBox* que aceita apenas datas:

```
using System;
using System.Windows.Forms;
. . .
// TextBox que valida datas
public class DataTextBox: TextBox {
 protected override void OnLeave(System.EventArgs e)
{
  base.OnLeave(e);
  try {
   Convert.ToDateTime(this.Text);
  }
  catch (FormatException) {
   // Segura o foco em caso de erro
   this.Focus();
  }
 }
}
```

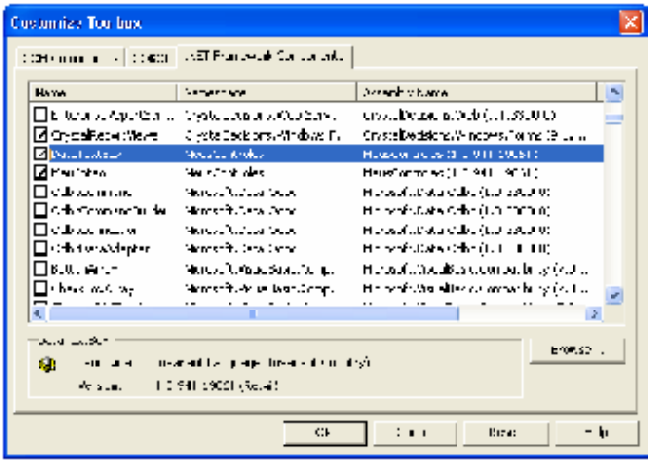
Se quiser, você pode colocar vários componentes em um mesmo fonte:

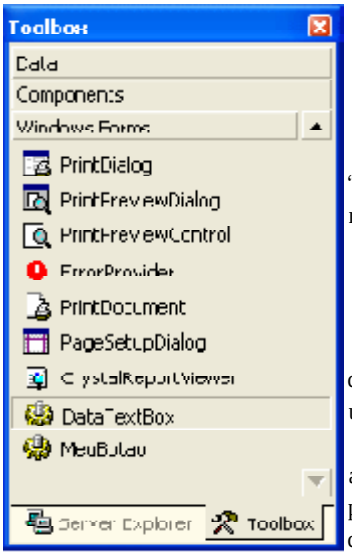


Já temos então um componente. Você pode instalá-lo no Visual Studio .NET clicando com o botão direito sobre o *ToolBox* e selecionando "Customize Toolbox...":



Clique em "Browse" e selecione a DLL com os componentes e aperte "Ok":





Clique novamente em "Ok" e observe o componente no *Toolbox*:

Você pode arrastar os componentes e colocá-los sobre um formulário.

Na segunda parte deste artigo veremos como adicionar propriedades e eventos aos componentes.

Acessando Oracle 9i via Internet através do Listener...

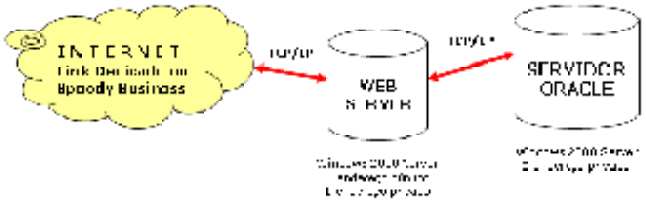
Escolhi para este mês um assunto que, tenho absoluta certeza, irá deliciá-los!

Vamos ensinar como acessar um banco 9i através da Internet utilizando o conceito de conexões via “listener”. Divirtam-se!!!

Uma questão muito presente nos dias de hoje e muito importante para a eficiência de qualquer organização, é ter a base de dados disponível na Internet, para que de qualquer localidade informações estratégicas possam ser acessadas.

Muito bem. A tecnologia Oracle oferece essa condição de forma contundente.

Vamos exemplificar um modelo básico, em que teríamos dois servidores: um servidor de web conectado de forma permanente à Internet e com um endereço IP público (numa placa de rede adicional ou diretamente num “hub” ou “switch”) e um servidor de dados.



Este modelo permite que tenhamos “resolvidas” várias questões. Primeiro, a questão segurança, já que o servidor de dados não estará diretamente “disponível” na Internet. Segundo a questão desempenho, já que todos os serviços “web” que eventualmente criarmos no “web server” (e geralmente queremos muitos desses serviços!) não estarão utilizando recursos necessários ao bom desempenho do banco de dados. Terceiro, a questão organização e controle, pois teremos servidores dedicados e especialmente configurados para tarefas específicas.

O papel dos dois servidores pode estar concentrado numa única máquina, mas para estruturas muito pequenas, tanto em tamanho da rede interna quanto em complexidade e porte da base de dados Oracle. E essa não poderá de forma alguma ser considerada uma solução profissional!

É fundamental que instalemos no servidor web um software que concentre:

- firewal básico;
- compartilhamento de conexão Internet;
- regras de acesso à Internet;
- mapeamento de portas.

Costumamos utilizar e recomendamos plenamente o Winroute, que reúne todas essas ferramentas e diversos serviços adicionais. As ferramentas similares disponíveis gratuitamente no Windows são muito rudimentares e incompletas e não costumam atender à exigências profissionais.

Conexões via "Web Server"

Uma das possibilidades que temos, principalmente se todas as "regras de negócio" estiverem escritas no banco, é conectarmos ao Oracle aplicações escritas em ASP, HTML ou qualquer outra linguagem, através de conexões ADO. É muito simples.

Neste caso, temos como característica só podermos nos conectar externamente ao banco através de chamadas do IIS (Internet Information Server) do Windows 2000 ou qualquer outro serviço "web server" que desejarmos.

É imprescindível neste caso, que tenhamos no servidor web um "serviço Oracle" (Host Name) apontando para o servidor Oracle local.

As conexões ADO estarão baseadas na existência deste recurso.

Conexões via Internet

No modelo de infra-estrutura exemplificado podemos também, se estivermos utilizando o Oracle 9i (Release 9.0.1.1.1 ou superior), nos conectarmos ao banco de dados através da Internet, configurando um "host name" em qualquer máquina externa apontando para o endereço público do servidor web (em versões anteriores do Oracle isto também é possível, mas exige que diversos "patchs" de atualização sejam rodados no Oracle e que outras configurações sejam feitas no Windows).

Para tanto, basta definirmos no Winroute que todas as chamadas à porta 1521 do servidor Web (se esta for a porta padrão do listener Oracle) sejam "mapeadas" para a porta 1521 do servidor Oracle.

Pronto! Se você estiver na Internet já poderá se conectar ao Oracle através da sua aplicação Delphi, do SQL Plus ou de qualquer outra forma de conexão que utilize o conceito de "listener".

Para o leitor mais atento isso pode significar um risco adicional: o de conexões indesejadas ao nosso banco de dados!!! E realmente esse risco está presente. Para evitá-lo, você pode configurar no Winroute um conjunto de máquinas (ou de endereços IP) que estarão autorizadas a utilizar o mapeamento de portas, definindo políticas de acesso.

Além disso, você pode ainda integrar a esse modelo outros servidores dedicados ao papel de firewall, utilizando plataformas Windows Server ou Linux.

É só isso! Simples não?

E mais um detalhe importante: a sua aplicação escrita em Delphi e largamente testada no Oracle 8i não precisará de nenhuma modificação para estar "compatível" com o Oracle 9i. Se essa ainda é uma de suas preocupações constantes, esqueça o problema!

DataSnap – Interfaces
Parte III

Por Anderson Haertel Rodrigues

Introdução

No artigo anterior foram demonstrados os principais padrões e protocolos existentes hoje no mercado que o DataSnap utiliza.

No artigo desse mês eu explicarei o que são *Interfaces*. Volto a frisar que usarei o **Delphi 6 Update Pack 2** para realizar nossos exemplos e explicações em geral.

Interfaces

A regra é clara: Uma *Interface* define um tipo que compreende métodos virtuais abstratos. Uma classe Delphi padrão (**TMyClass = class(TObject)**) tem de ser herdada apenas de uma classe, por que, o Delphi não dá suporte a Múltipla(s) Herança(s). *Interfaces* existem para contornar esse problema (falando em Programação Orientada a Objeto) e ainda trabalhar com Objetos Distribuídos.

As *Interfaces* foram implementadas no Delphi 3 justamente para o Delphi dar suporte ao padrão COM da Microsoft®, mas, não significa que temos de conhecer COM para usarmos *Interfaces* no Delphi. Assim como todas classes Delphi no seu mais Baixo nível herdam de TObject (unit System.pas), todas as *Interfaces* herdam de: **IInterface** (unit System.pas) conforme listado abaixo:

```
IInterface = interface
 ['{00000000-0000-0000-C000-000000000046}']

function QueryInterface(const IID: TGUID; out Obj):
 HResult; stdcall;
function _AddRef: Integer; stdcall;
function _Release: Integer; stdcall;

end;
```

Listagem 1: Definição de IInterface.

Todos os **Objetos que servirão como classe base para a implementação de COM implementam IUnknown** (unit System.pas) conforme listado abaixo:

```
IUnknown = IInterface;
```

Listagem 2: Definição de IUnknown.

Nota: Até a versão 5 do Delphi, todas as Interfaces descendiam de IUnknow, então se nós quiséssemos implementar uma Interface que não estivesse ligada a COM, não tínhamos como! Necessitaríamos herdar da classe TInterfacedObject para criarmos uma Classe não COM. Na versão 6 do Delphi, a Borland decidiu criar uma interface Object Pascal não ligada a COM.

A function **_AddRef** e **_Release** gerenciam a contagem de referência do objeto. Quando a contagem de referência do objeto chega a zero (0), _Release libera o objeto. Quem realmente controla a contagem de referência das *Interfaces* é o próprio compilador do Delphi. O Controle é o mesmo para Strings e Arrays Dinâmicos.

A function **QueryInterface** retorna a referência do objeto, se o objeto implementou tal *interface*.

COM, CORBA e SOAP

Como já citado, *Interfaces* são necessárias para implementação e uso de Objetos COM, CORBA e SOAP. *Interfaces* estão cada vez mais ligadas ao nosso dia-a-dia e vem com mais força ainda na implementação de **WebServices**.

Cada definição de *Interface* necessita de um GUID, que já explicado, pode ser gerado pelo Delphi através de **CTRL+SHIFT+G** no editor de códigos do Delphi.

Vamos agora implementar e exemplificar o uso de *Interfaces*

Herança Múltipla

Como já citado, herança de Classe força a Classe Filha a armazenar todos os campos que são declarados na Classe PAI. Não tem jeito! Em herança de classes, podemos acrescentar funcionalidades a Classe Filha, mas, não podemos remover funcionalidades da Classe PAI, o máximo que podemos na Classe Filha, é tornar as coisas invisíveis. Se não quisermos tal funcionalidade na Classe Filha, teremos de ir ao Ancestral imediato a Classe PAI e verificar se o mesmo implementou tal funcionalidade, caso tenha implementado, teremos de ir Classe Ancestral por Classe Ancestral até chegarmos na Classe que não implementa tal funcionalidade. Nesse processo de retorno de Classes Ancestrais podemos e, com certeza perderemos implementações vitais, dos quais, nos farão falta na Classe Filha. É um problema!

Vejam bem, a Herança de Classe é uma ferramenta eficaz (graças a Deus que a mesma existe) para a reutilização de código. O Problema (ou não) é o Delphi não dar suporte a Herança(s) Múltipla(s). Um debate sobre os benefícios ou não de Herança(s) Múltipla(s) está fora dos planos desse artigo.

Square x Polygon x Rectangle = Shape

A *Interface* IShape seria a *Interface* base para todas as *Interfaces* IPolygon, IRectangle e ISquare envolvidas na estrutura. TSquare necessita armazenar todos os seu lados, TRectangle necessita armazenar dois tamanhos e TPolygon necessita armazenar muitos lados e vértices. No esquema apresentado acima não nos interessa guardar dados pertinentes de uma forma (Shape) para a outra, o que nos interessa é termos uma *Interface* e uma **Classe Base** que armazena todos campos que façam sentidos em todas as formas apresentadas. Qual a solução? É separar a herança de Tipo (**Interfaces**) da Herança de Classe, como já citado.

```
type

IShape = interface
['{5C3AE4B8-B50E-44C6-A135-6FF00E44A270}']

   procedure Draw(Canvas: TCanvas);
   procedure SetPosition(const Value: TPoint);
   function GetPosition: TPoint;
   property Position: TPoint read GetPosition write
   SetPosition;

end;

IPolygon = interface(IShape)
['{ACB90E43-04E6-4A7D-A782-438B5C201E28}']

   function NumVertices: SmallInt;
   function NumSides: SmallInt;
   function SetLenght(const Index: SmallInt): TPoint;
   function Vertex(Index: SmallInt): TPoint;

end;

IRectangle = interface(IPolygon);
['{31224B6F-4488-4280-BA93-7E5DDA71F2ED}']
end;

ISquare = interface(IRectangle);
['{A7787A55-8066-458F-A3D1-538E86DF8774}']

   function Side: SmallInt;

end;

TShapeBase = class(TInterfacedObject, IShape)
private
  { Private declarations }

  FPosition: TPoint;

  function GetPosition: TPoint;

  procedure SetPosition(const Value: TPoint);

protected
  { Protected declarations }

public
  { Public declarations }

   constructor Create; override;

   procedure Draw(Canvas: TCanvas); virtual;
abstract;
   property Position: TPoint read GetPositon write
SetPosition;

published
  { Published declarations }
```

```
end;

TPolygon = class(TShapeBase, IPolygon)
private
  { Private declarations }

  FVertices: array of TPoint;

protected
  { Protected declarations }

public
  { Public declarations }

   procedure Draw(Canvas: TCanvas); override;

   function NumVertices: SmallInt;
   function NumSides: SmallInt;
   function SideLenght(const Index: SmallInt):
 SmallInt;
   function Vertex(Index: SmallInt);

published
  { Published declarations }

end;

TRectangle = class(TShapeBase, IPolygon, IRectangle)
private
  { Private declarations }

  FRect: TRect;
protected
  { Protected declarations }

public
  { Public declarations }

   procedure Draw(Canvas: TCanvas); override;

   function NumVertices; SmallInt;
   function NumSides: SmallInt;
   function SideLenght(const Index: SmallInt):
 SmallInt;
   function Vertex(Index: SmallInt);

published
  { Published declarations }

end;
```

```
TSquare = class(TShapeBase, IPolygon, IRectangle,
 ISquare)
private
  { Private declarations }

  FSide: SmallInt;

protected
  { Protected declarations }

public
  { Public declarations }

   procedure Draw(Canvas: TCanvas); override;

   function Side: SmallInt;
   function NumVertices: SmallInt;
   function NumSides: SmallInt;
   function SideLenght(const Index: SmallInt):
 SmallInt;
   function Vertex(Index: SmallInt);

published
  { Published declarations }

end;
```

Listagem 3: A estrutura de *Interface* e implementação das classes.

Não é necessário implementar o código acima! É necessário um comentário para um entendimento maior por parte do leitor. A Classe Base para as classes apresentadas acima é a Classe TShapeBase, que implementa a *Interface* IShape. Se olharmos com mais calma e atenção à estrutura acima, veremos que nem todas as classes (TPolygon, TRectangle e TSquare) implementam os métodos disponíveis nas 03 (Três) *interfaces* criadas. Um exemplo claro é a classe TRectangle, que herda de TShapeBase e implementa IPolygon e IRectangle, por que, são os métodos das duas *interfaces* citadas que fazem sentido para a classe TRectangle. Da mesma forma acontece com a classe TPolygon. A única classe que implementa todas as *Interfaces* é a classe TSquare.

Delegação

Uma classe pode delegar a implementação de uma *interface* para uma propriedade, desde que esta use a diretiva **implements** e o valor da propriedade precisa ser do tipo da *interface* e ou da classe do qual você deseja implementar. O método não pode ser dinâmico, apenas virtual e não pode ser uma propriedade do tipo Array. A sintaxe de **implements** é a seguinte:

propriedade read Getter implements *interfaces...;*

Exemplo: Podemos exemplificar **implements** da seguinte forma:

```
type

  IMyIntf1= interface
    procedure P1;
    procedure P2;

  end;

   { Declaração Forward. }
   IMyIntf2 = interface;

  TMyClass = class(TObject, IMyIntf1)
    FMyInterface: IMyIntf2;
    property MyInterface: IMyIntf2 read FMyInterface
 implements IMyIntf2;

  end;

   implementation

   IMyIntf2 = interface
    procedure P3;

  end;

end.
```

Listagem 4: Demonstra o uso da diretiva implements

O código acima demonstra o uso da diretiva **implements**. O que fizemos acima foi, delegar a implementação da propriedade para a interface IMyIntf2, que está na diretiva implementation.

Quando temos, tipos, sub-rotinas e ou variáveis nesta seção, as mesmas são privadas a essa unidade.

O código acima foi uma delegação para um tipo *interface* se o mesmo fosse um tipo de classe, teríamos que tomar o seguinte cuidado:

- O tempo de vida do objeto. Para contornar as dores de cabeças que poderão vir, temos que fazer com que o Getter crie um novo objeto toda vez que for chamado.

Se a sua declaração necessitar implementar mais de uma *Interface* separe as mesmas por virgula.

Considerações finais:

- O Compilador do Delphi não trata as *interfaces* por seu nome e sim pelo seu ID;

- Você pode passar o nome da *Interface* para o método QueryInterface que o Delphi automaticamente transforma seu valor no ID da *Interface;*

- A Classe TObject tem dois métodos interessantes para o retorno de informações nas *Interfaces* suportada pelo Objeto: GetInterfaceTable() e GetInterfaceEntry(), apenas saliento que a última você raramente usará, por que, é usada internamente pelo Delphi no método GetInterface();

- O Incremento de referência da *Interface* é conseguido usando o operador **as** e com o método QueryInterface();

- A seguinte declaração faz com a *interface* herde de **IInterface**:

IMyIntf = interface, a mesma é análoga a TMyClass = Class, do qual a classe está sendo herdada de TObject.

Conclusão

Mostrei nesse artigo o conceito e para que serve *Interfaces.* Como já citado, *Interfaces* já fazem parte do nosso dia-a-dia e farão mais ainda com a chegada dos **WebServices**. Foi demonstrado também que, *Interfaces* não está ligada a apenas OD (Objetos Distribuídos), pois, com a estrutura de *Interfaces* e classes demonstradas nesse artigo, vimos que a mesma também é de extrema importância na Programação Orientada a Objetos (OOP).

No próximo artigo daremos início à parte de criação de servidores de aplicação no protocolo COM+. Além de criar o servidor, eu mostrarei o que é e para que serve cada opção na criação do **Remote DataModule** (RDM).

Sucesso e Saúde a todos!

Um Abraço.

Anderson Haertel Rodrigues

Consultor em desenvolvimento de Sistemas Client/Server-MultiCamadas/DataSnap.
Oferece treinamentos para empresas em: InterBase, SQL-Server, Delphi e COM+.
anderson.hr@bol.com.br - anderson.hr@zipmail.com.br.
Florianópolis - Santa Catarina - Brasil.

Homologando seu Software para realizar Transferência Eletrônica de Fundos

Por Claudenir C. Andrade

Em artigos anteriores, comentamos sobre a necessidade de uma vez que seu aplicativo esteja totalmente preparado para fazer TEF, falando com a dll de comunicação e o gerenciador padrão de cada bandeira, existe a necessidade de realizar uma homologação ou testes no software em duas entidades homologadoras de TEF, a Software Express e a Seven Pdv, ambas em São Paulo.

Nestes testes são realizadas todas as operações que envolvem a Transferência Eletrônica de Fundos, ou seja, não será certificado seu software se esta baixando estoque ou não, se esta "bonitinho" ou não, a questão é homologar para certificar-se que as operações de transação eletrônica realmente estão acontecendo de acordo com os padrões de segurança estabelecidos pelas bandeiras. O que é altamente aceitável pois com a nova lei da Obrigatoriedade do TEF nas impressoras fiscais, nossos softwares passam a assumir uma pesada responsabilidade que é débito ou credito de valores monetários de nossos clientes.

Nestes passos homologatórios existem vários testes que são realizados, entre eles se a impressora fiscal está desligada e com o Micro (PC) desligado. Como é feito isso:

1) É ordenada a impressão de um cupom fiscal e logo após uma Transação TEF, quando o software começa a imprimir o comprovante TEF na Impressora fiscal, exatamente neste momento, o ECF é desligado e você necessita controlar tudo isso em seu software.

a) Necessita identificar que o ECF está desligado
b) Necessita controlar que ao ligar o ECF novamente você deverá reiniciar a impressão do TEF em um Relatório Gerencial
c) E finalmente necessita controlar que caso o ECF não seja ligado novamente, a operação deverá ser Cancelada

2) É ordenada novamente a impressão de um cupom fiscal e logo após uma transação ETF, quando o software começa a imprimir o comprovante TEF na Impressora o Micro (PC) é desligado, isso mesmo, e você deverá controlar:

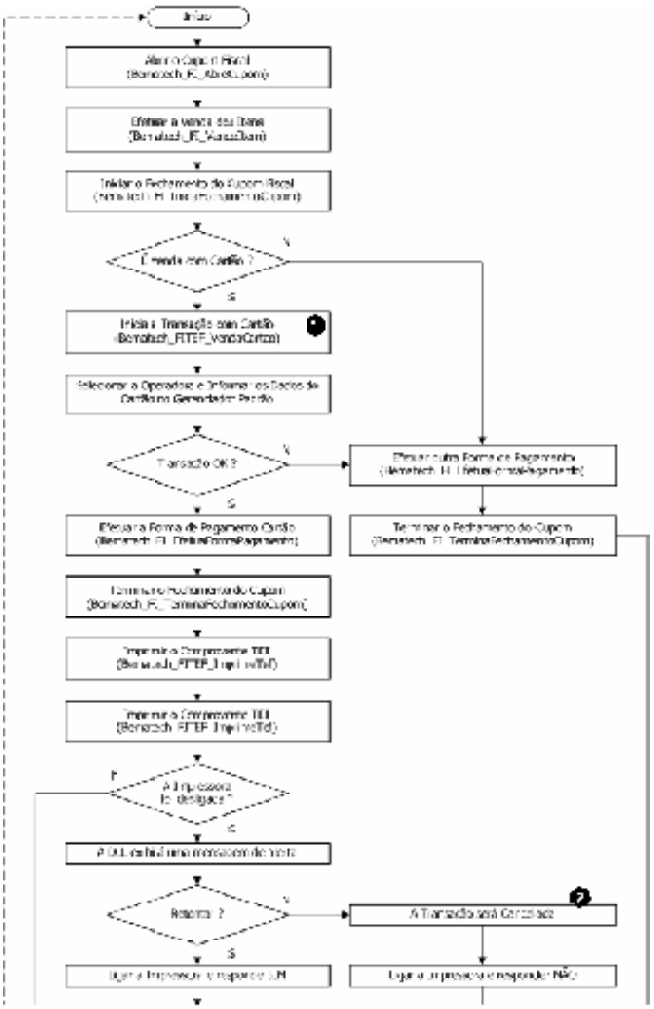
a) Necessita Identificar que o micro foi desligado
b) Necessita identificar que ficou uma operação pendente
c) Necessita cancelar esta operação e fechar o comprovante que ficou aberto.

Felizmente a dll de comunicação economiza uma boa parte dos controles que você necessita fazer em seu software, como identificar se a impressora esta desligada ou não, reimprimir a transação em um relatório Gerencial, etc..

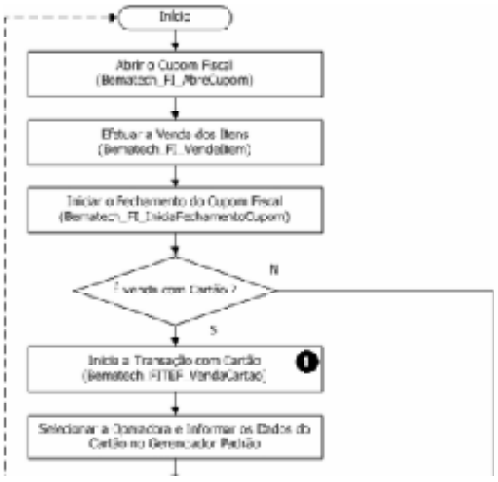
Nada melhor que "fluxogramas" para entendermos então como fazer isso, estes fluxos você pode encontrar em nosso site (www.bematech.com.br) na área de download, impressoras fiscais, poderá baixar o arquivo de Help da dll BamFi32.dll, que é nossa dll de comunicação de alto nível.

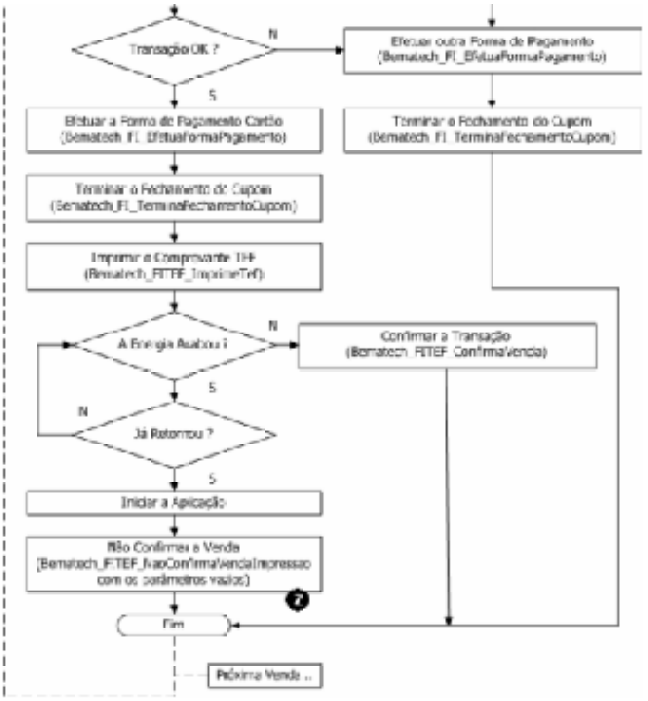
Bem, Observe os Fluxogramas abaixo:

Cancelar e controlar o ECF quando o mesmo é desligado no momento da impressão do TEF



Cancelar e controlar o MICRO (PC) é desligado no momento da impressão do TEF





Sendo assim, para cada uma destas operações você terá apenas um comportamento diferenciado de seu software, possibilitando total controle das operações TEF.

Observe nos fluxos que cada função da DLL BemaFi32.dll elimina o trabalho que a software house e seu software terá de controlar quantas linhas foram impressas, etc... Controlar se o ECF esta ligado e seus sinais de porta serial etc.., tudo isso está encapsulado em nossa dll de comunicação, a BemnaFi32.dll.

Sendo assim, agora que você está preparado para a homologação, Mão na Massa com a BemaFi32.dll!!

Claudenir C. Andrade

É gerente de desenvolvimento de software na Bematech e autor do livro "Automação Comercial com C#, VB.NET e Delphi 6" lançado pela editora Érica – www.erica.com.br.

Ele também pode ser contatado pelo e-mail:

claudenir@bematech.com.br

Trabalhando com string no Delphi 6

A cada nova versão o Delphi traz mais novidades. A nova unit StrUtils traz novas rotinas para o trabalho com strings, complementando àquelas já existentes na unit SysUtils do Delphi 5.

A unit SysUtils tem várias rotinas para manipulação de strings. Você provavelmente está familiarizado com as funções UpperCase, LowerCase, CompareStr, CompareText, IntToStr, IntToHex, StrToInt entre outras. Mas você talvez não tenha notado as rotinas QuotedStr e AnsiQuotedStr que adicionam aspas no inicio e no final de uma string, e em contrapartida a rotina AnsiExtractQuotedStr que retira as aspas. A nova função AnsiDequotedStr torna mais fácil usar o último com strings ordinárias.

```
function QuotedStr(const S: string): string;
function AnsiQuotedStr(const S: string; Quote: Char):
    string;
function AnsiExtractQuotedStr(var Src: PChar; Quote:
    Char): string;
function AnsiDequotedStr(const S: string; AQuote:
    Char): string;
```

Existe também a rotina AdjustLineBreaks que modifica um simples line feeds ou carriage returns em uma combinação de dois caracteres usados no Windows. No Delphi 6 você também pode ir por outro caminho convertendo a combinação, ou um simples carriage returns para simples line feeds.

```
function AdjustLineBreaks(const S: string;
 Style: TTextLineBreakStyle =
  {$IFDEF LINUX} tlbsLF {$ENDIF}
  {$IFDEF MSWINDOWS} tlbsCRLF {$ENDIF}): string;
```

A rotina StrToIntDef é como um StrToInt. A diferença é que se for informado um valor inválido para a variável S a função retorna o número passado na variável Default.

```
function StrToIntDef(const S: string; Default:
  Integer): Integer;
```

Para formatar um texto em uma área limitada, você pode usar a rotina de WrapText. Esta rotina faz a pesquisa da linha fornecida do texto e, se for mais longo do que o tamanho máximo especificado ela substitui a última ocorrência antes daquela posição com o texto em BreakStr. A versão mais simples quebra em espaços, em tabs, ou em hífens. Note que esta rotina não quebra as substrings entre aspas.

```
function WrapText(const Line, BreakStr: string; const
     BreakChars: TSysCharSet; MaxCol: Integer):
      string; overload;
function WrapText(const Line: string; MaxCol: Integer
     = 45): string; overload;
```

Novidades no SysUtils

Diversas novas rotinas foram incluídas na unit SysUtils do Delphi 6. Dentre elas inclui a constante WideString, rotinas básicas de versão WideString, sobrecarga de funções Trim para WideString e rotinas de formação WideString.

```
const
   EmptyWideStr: WideString = '';
   NullWideStr: PWideString = @EmptyWideStr;

function WideUpperCase(const S: WideString):
WideString;
function WideLowerCase(const S: WideString):
WideString;
function WideCompareStr(const S1, S2: WideString):
Integer;
```

```
function WideSameStr(const S1, S2: WideString):
Boolean;
function WideCompareText(const S1, S2: WideString):
Integer;
function WideSameText(const S1, S2: WideString):
Boolean;

function Trim(const S: WideString): WideString;
overload;
function TrimLeft(const S: WideString): WideString;
overload;
function TrimRight(const S: WideString): WideString;
overload;

function WideFormat( const Format: WideString;
 const Args: array of const): WideString;
procedure WideFmtStr(var Result: WideString; const
Format: WideString; const Args: array of const);
function WideFormatBuf(var Buffer; BufLen: Cardinal;
const Format; FmtLen: Cardinal; const Args: array of
const): Cardinal;
```

As rotinas de conversão existentes de outro tipo básico para strings tem sido melhoradas com versões que pegam valores default – retornando-o se uma exceção for apresentada de qualquer forma. Existe também rotina correspondente ao Try que executa a conversão e retorna True se teve sucesso e False se não teve sucesso.

Como um efeito colateral, ele retorna um valor convertido, via um parâmetro.

```
function StrToFloatDef(const S: string; const
     Default: Extended): Extended;
function TryStrToFloat(const S: string; out Value:
     Extended): Boolean; overload;
function TryStrToFloat(const S: string; out Value:
     Double): Boolean; overload;
function TryStrToFloat(const S: string; out Value:
     Single): Boolean; overload;

function StrToCurrDef(const S: string;  const
     Default: Currency): Currency;
function TryStrToCurr(const S: string; out Value:
     Currency): Boolean;

function StrToDateDef(const S: string; const Default:
     TDateTime): TDateTime;
function TryStrToDate(const S: string; out Value:
     TDateTime): Boolean;
function StrToTimeDef(const S: string; const Default:
     TDateTime): TDateTime;
function TryStrToTime(const S: string; out Value:
     TDateTime): Boolean;

function StrToDateTimeDef(const S: string; const
     Default: TDateTime): TDateTime;
function TryStrToDateTime(const S: string; out Value:
     TDateTime): Boolean;
```

Você pode agora converter facilmente entre boolean e string com o default e versões Try descritas anteriormente. A conversão depende dos arrays TrueBoolStrs e FalseBoolStrs. Você os preenche com os valores apropriados e qualquer comparação de string retorna True ou False. Para converter os valores de um boolean para string use a primeira entrada desses arrays.

```
var
   TrueBoolStrs: array of String;
   FalseBoolStrs: array of String;

 const
   DefaultTrueBoolStr = 'True'; // DO NOT LOCALIZE
   DefaultFalseBoolStr = 'False'; // DO NOT LOCALIZE

function StrToBool(const S: string): Boolean;
function StrToBoolDef(const S: string; const Default:
     Boolean): Boolean;
function TryStrToBool(const S: string; out Value:
     Boolean): Boolean;
function BoolToStr(B: Boolean; UseBoolStrs: Boolean =
     False): string;
```

Comparação

Diversas funções convenientes que estão disponíveis para comparação de valores de duas ou mais strings. Elas vêm em duas formas: case sensitive e not case sensitive. A primeira tem nomes terminados em str, enquanto as outras têm o nome terminado em Text. O primeiro conjunto de funções olha para uma substring em qualquer lugar em uma string. Cada um retorna um simples True ou False. De qualquer forma, aquela ordem dos parâmetros é revertida na rotina Contains como comparado a outras e a função AnsiPos.

```
function AnsiContainsText(const AText, ASubText:
     string): Boolean;
function AnsiStartsText(const ASubText, AText:
     string): Boolean;
```

```
function AnsiEndsText(const ASubText, AText: string):
     Boolean;
function AnsiContainsStr(const AText,  ASubText:
     string): Boolean;
function AnsiStartsStr(const ASubText,  AText:
     string): Boolean;
function AnsiEndsStr(const ASubText,  AText: string):
     Boolean;
```

Você também pode comparar uma simples string com um array de valores, retornando um simples flag indicando sua presença ou seu índice dentro do array. O index conta a partir de zero, e – 1 quando o valor não existe. Novamente, o Str é case-sensitive, enquanto o Text não é.

```
function AnsiMatchText(const AText: string; const
     AValues: array of string): Boolean;
function AnsiIndexText(const AText: string; const
     AValues: array of string): Integer;

function AnsiMatchStr(const AText: string; const
     AValues: array of string): Boolean;
function AnsiIndexStr(const AText: string; const
     AValues: array of string): Integer;
```

Alterando

Trocar todas as ocorrências de uma substring em uma string com outro valor. Internamente eles usam a rotina StringReplace da unit SysUtils, especificando a troca total como um flag e ignorando o caso quando apropriado.

```
function AnsiReplaceText(const AText, AFromText,
     AToText: string): string;
function AnsiReplaceStr(const AText, AFromText,
     AToText: string): string;
```

Reverte a ordem dos caracteres na string com a função ReverseString.

```
function ReverseString(const AText: string): string;
```

DupeString concatena a string fornecida um determinado número de vezes e retorna o resultado.

Você não recebe um erro se o número de repetições é zero ou menor, mas você não recebe qualquer saída para ambos.

```
function DupeString(const AText: string; ACount:
     Integer): string;
```

StuffString troca a seção nomeada de uma string existente (AStart e ALength) com a nova substring. Você recebe alguns efeitos interessantes se o tamanho for negativo – você começa repetindo alguns caracteres mais adiantados na string original que segue a substituição.

```
function StuffString(const AText: string; AStart,
  ALength: Cardinal; const ASubText: string): string;
```

Seleção

Para escolher uma entrada aleatória de um array de strings, cada um com uma possibilidade igual, use a rotina RandomFrom. Note que esta é uma função sobrecarregada que pode também fazer exame de disposições de números inteiros ou de ponto flutuante.

```
function RandomFrom(const AValues: array of string):
     string; overload;
```

Outra função sobrecarregada, IfThen fornece uma função em linha para strings. Ela avalia a expressão booleana e retorna a primeira string do parâmetro se for verdadeiro, e a segunda se for falso. Esta função pode ser embutida em uma grande expressão e troca uma instrução if para executar o cálculo.

```
function IfThen(AValue: Boolean; const ATrue: string;
     AFalse: string = ''): string; overload;
```

Se você precisa de um acesso fácil para as seções de string, você pode usar as seguintes rotinas para extrair o inicio e o fim ou alguma coisa entre eles. Cada um deles usa internamente a função Copy. De fato, o MidStr é basicamente um Copy renomeado.

```
function LeftStr(const AText: string; const ACount:
     Integer): string;
function RightStr(const AText: string; const ACount:
     Integer): string;
function MidStr(const AText: string; const AStart,
     ACount: Integer): string;
```

Procurando

A função SearchBuf deixa você procurar um texto arbitrário, ou parte disso, por uma informação em particular. As opções permitem que você pesquise para cima ou para baixo a partir da posição corrente, determina se o caso foi considerado compatível, e especifica se somente as palavras completas serão procuradas. A direção de pesquisa default é do inicio para o final do arquivo. Como você pode ver a partir da constante WordDelimiters,

qualquer caractere com exceção de um alfanumérico denota um limite de palavra.

Informe os valores SelStart e SelLength para saltar sobre este ponto do buffer. Neste caso, começamos a pesquisa após ou antes da seleção baseada na presença da opção soDown.

O valor de retorno da função é um ponto para o inicio da primeira ocorrência da string de pesquisa dentro da seção denominada de buffer, ou nil se ele não puder ser encontrado lá.

```
const
  { Default word delimiters are any character
    except the core alphanumerics. }
  WordDelimiters: set of Char =
    [#0..#255] - ['a'..'z','A'..'Z','1'..'9','0'];
type
  TStringSeachOption =
    (soDown, soMatchCase, soWholeWord);
  TStringSearchOptions =
    set of TStringSeachOption;

function SearchBuf(Buf: PChar; BufLen: Integer;
SelStart, SelLength: Integer; SearchString: String;
  Options: TStringSearchOptions = [soDown]): PChar;
```

Conclusão

O Delphi tem muita funcionalidade embutida. Enquanto o suporte da manipulação de strings sempre esteve presente no Delphi, a última versão adiciona muito mais e move uma parte para outra unit, StrUtils.

Esta matéria foi uma pequena introdução sobre as novas potencialidades, e uma revisão de algumas das mais antigas, o objetivo foi deixar você ciente destas rotinas de modo que as possa usar dentro de seus próprios programas.

Detectando URLs no RichEdit

Você já deve ter visto esta característica em outros softwares e desejou que sua aplicação fizesse isto também. Bem, aqui está uma matéria que realmente prenderá seus olhos. Você já deve ter notado que o Microsoft Word tem o reconhecimento de URLs. Neste artigo vamos mostrar como criar esta característica no componente RichEdit.

O RichEdit e a detecção da URL

Se você fizer uma pesquisa no seu computador pelos arquivos riched*.dll, você encontrará dois arquivos: RICHED32.DLL e RICHED20.DLL que estão nos diretórios \windows\system ou winnt\system32 dependo da versão do seu Windows. O arquivo RICHED32.DLL implementa as características da versão 1.0 e o RICHED20.DLL implementa uma nova característica ao RichEdit.

O componente incluído no Delphi não tem a implementação de detecção de URL. Se você olhar o código fonte do RichEdit (comctrls.pas), o Delphi ainda usa o RichEdit versão 1.0. Isto é triste porque o controle Microsoft RichEdit versão 2.0 não tem a detecção de URL e muitas outras características.

Como podemos reconstruí-la. Nós temos esta tecnologia.

Portanto existe apenas uma coisa a fazer. Refazer o controle baseado no novo RichEdit 2.0.

Primeiro vamos copiar o arquivo comctrls.pas para um novo arquivo chamado Riched20.pas. Remova todas as classes, menos as classes que pertencem à classe TCustomRichEdit. Renomeie a classe para TExCustomRichEdit. Agora comece a fazer as suas implementações na versão 2.0 do RichEdit.

Na procedure CreateParams, modifique o conteúdo da constante RichEditModuleName de Riched32.DLL para Riched20.dll, como mostrado na listagem 1. Modifique também a procedure CreateSubClass, desde então ele estava usando a versão antiga. Existem duas classes fornecidas: a ANSI e a UNICODE que contém as constantes RICHEDIT_CLASSA e RICHEDIT_CLASSW respectivamente. Nós vamos usar a versão ANSI da classe RichEdit, que é a RICHEDIT_CLASSA.

```
procedure TExtCustomRichEdit.CreateParams
        (var Params: TCreateParams);
 const
   RichEditModuleName = 'RICHED20.DLL';

   HideScrollBar : array[Boolean] of DWORD =
            (ES_DISABLENOSCROLL, 0);
   HideSelections: array[Boolean] of DWORD =
            (ES_NOHIDESEL, 0);

 begin

   if FRichEditModule = 0 then
   begin
      FRichEditModule :=
                LoadLibrary(RichEditModuleName);
      if FRichEditModule <= HINSTANCE_ERROR then
                FRichEditModule := 0;
   end;

   inherited CreateParams(Params);
   CreateSubClass(Params, RICHEDIT_CLASSA);
```

```
 with Params do
  begin
    Style := Style or HideScrollBar[HideScrollBars]
or
      HideSelections[HideSelection];
    WindowClass.style := WindowClass.style and not
              (CS_HREDRAW or CS_VREDRAW);
  end;
end;
```

Listagem 1: A procedure CreateParams no novo componente RichEdit baseado no Microsoft RichEdit 2.0

Processando a detecção da URL

Em ordem para o controle receber a mensagem EN_LINK, nós vamos incluir a mensagem ENM_LINK quando enviar o EM_SETEVENTMASK, veja a listagem 2. Também quando criar a janela, nós enviamos a mensagem EM_AUTOURLDETECT a qual ativa o efeito CFE_LINK.

Este efeito modifica a cor do texto URL e deixa o texto sobrescrito.

```
procedure TExtCustomRichEdit.CreateWnd;
 var
   Plain, DesignMode, WasModified: Boolean;
 begin
   WasModified := inherited Modified;

   inherited CreateWnd;
   if (SysLocale.FarEast) and
        not (SysLocale.PriLangID = LANG_JAPANESE)
then
    Font.Charset := GetDefFontCharSet;
 // Added the ENM_LINK to receive EN_LINK message

   SendMessage(Handle, EM_SETEVENTMASK, 0,
     ENM_CHANGE or ENM_SELCHANGE or
     ENM_REQUESTRESIZE or
     ENM_PROTECTED or ENM_LINK);
 // Activate the URL Detection feature
   SendMessage(Handle, EM_AUTOURLDETECT,
                          Ord(FURLDetect), 0);

 SendMessage(Handle, EM_SETBKGNDCOLOR, 0,
                         ColorToRGB(Color));
   if FMemStream <> nil then
   begin
     Plain := PlainText;
     FMemStream.ReadBuffer(DesignMode,
                          sizeof(DesignMode));
     PlainText := DesignMode;
     try
       Lines.LoadFromStream(FMemStream);
       FMemStream.Free;
       FMemStream := nil;
     finally
       PlainText := Plain;
     end;
   end;

   Modified := WasModified;
end;
```

Listagem 2: A procedure CreateWnd.

Quando o cursor do mouse for movido para o texto da URL, o controle RichEdit recebe uma notificação EM_LINK. Neste ponto, o controle pode mudar o cursor do mouse, indicando ao usuário que ele pode clicar na URL.

Veja a listagem 3.

```
procedure TExtCustomRichEdit.CNNotify
                  (var Message: TWMNotify);
 type
   PENLink = ^TENLink;
  begin
   with Message do
     case NMHdr^.code of
       EN_SELCHANGE:
                 SelectionChange;
       EN_REQUESTRESIZE:
                  RequestSize(PReqSize(NMHdr)^.rc);
       EN_SAVECLIPBOARD:
         with PENSaveClipboard(NMHdr)^ do
           if not SaveClipboard(cObjectCount, cch)
                then Result := 1;
       EN_PROTECTED:
         with PENProtected(NMHdr)^.chrg do
           if not ProtectChange(cpMin, cpMax) then
                  Result := 1;
  // EN_LINK message being received to respond to it
       EN_LINK:
        begin
         Windows.SetCursor
              (Screen.Cursors[crHandPoint]);
        if PEnLink(NMHdr)^.msg = WM_LBUTTONDOWN then
          begin
 // set the selection
```

```
            SendMessage(Handle, EM_EXSETSEL, 0,
                  Longint(@PEnLink(NMHdr)^.chrg));
// send it to windows to open
            ShellExecute(handle, 'open',
                  PChar(GetSelText), nil, nil,
                        SW_SHOWNORMAL);
          end;
       end;
    end;
end;
```

Quando notificado, a mensagem EM_LINK inclui a estrutura NMHdr que aponta para a estrutura ENLINK, como mostrado aqui.

```
ENLINK = record
   nmhdr: TNMHdr;
   msg: UINT;
   wParam: WPARAM;
   lParam: LPARAM;
   chrg: TCharRange;
end;
```

Você pode distribuir a estrutura NMHdr com um ponteiro para a estrutura ENLINK para receber campo msg que contém a mensagem WM_LBUTTONCLICK.

Além disso a variável chrg hospeda a localização e o tamanho da URL dentro do texto.

```
_charrange = record
   cpMin: Longint;
   cpMax: LongInt;
end;
```

Lançando a URL

Para configurar a seleção, nós vamos usar a variável chrg para enviar a mensagem EM_EXSETSEL para retornar o texto da URL.

```
SendMessage(Handle, EM_EXSETSEL, 0,
Longint(@PEnLink(NMHdr)^.chrg));
```

Para rodar a aplicação que está apropriada a URL, nós vamos usar a função ShellExecute:

```
ShellExecute(handle, 'open', PChar(GetSelText), nil,
nil, SW_SHOWNORMAL);
```

Configurando a propriedade de detecção da URL

Nós vamos criar uma propriedade URLDetect com o novo controle TExRichEdit.

```
property URLDetect : boolean
   read FURLDetect
   write SetURLDetect
   default FALSE;
```

A procedure SetURLDetect foi escrita apenas para que quando a propriedade URLDetect seja configurada, ela chama a função RecreateWnd que recria a procedure CreateWnd, veja a listagem 2. Isto atualiza o controle RichEdit corrente para desligar ou ligar a detecção da URL.

```
procedure TExtCustomRichEdit.SetURLDetect(
   Value: boolean);
begin
   if URLDetect <> Value then
     begin
       FURLDetect:= Value;
       RecreateWnd;
     end;
end;
```

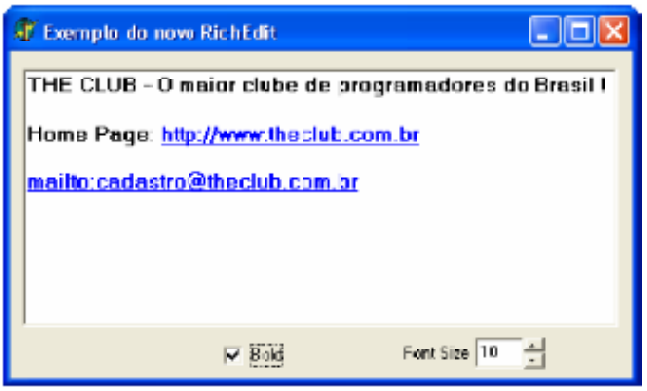


Figura 1: Utilizando o novo RichEdit

Conclusão:

Finalmente você pode ver na figura 1 um exemplo de uma aplicação implementada com o novo RichEdit 2.0. Ela não suporta apenas URLs, mas também file: ,mailto:, ftp:, https:, gopher:, nntrp:, properto:, telnet:, news:, wais:

Por implementar um componente construído no RichEdit 2.0 da Microsoft, nós fomos capazes de usar a característica de deteção de URL. Entretanto, olhando a lista das características incluídas na versão 2.0, é muito provável que você encontre outras características que gostaria de usar. Você pode fazer o download do exemplo no endereço www.theclub.com.br/revista/novorichedit.zip, mas não esqueça que antes de testar o exemplo você deve instalar o novo componente ou seja, o arquivo Riched20.pas.

Perguntas & Respostas

Pergunta: Quando tento executar um programa compilado com o Delphi 6.0 no Windows NT 4 apresenta erro "Não foi possível localizar o ponto de entrada do procedimento VarNot na biblioteca vínculo dinâmico OleAut32.dll". Como resolver?

Resposta: Para solucionar este problema o Sr. deverá rodar um Update Pack em seu Windows:

Windows NT/2000: Requer Windows NT 4.0 SP4 ou superior.
Windows 95/98: Requer Windows 98 (ou Windows 95 com DCOM 1.2).

Dúvida enviada por Cosis Computação & Sistemas, Franca/SP

Pergunta: Tenho uma aplicação desenvolvida em Delphi6+dbExpress+Firebird e quando declaro a unit MidasLib na lista de uses, as exceções retornadas pelo banco de dados são apresentadas, porém sem a mensagem de erro na caixa de diálogo, aparecendo somente o botão "Ok". Porém, compilando sem a MidasLib na lista de uses funciona corretamente. Como resolver o problema?

Resposta: Na verdade isso ocorre devido a um bug no Delphi, bug esse que pode ser resolvido atualizando o arquivo MidasLib.DCU e incluindo o arquivo MidasLib.RES no diretório LIB do Delphi6. Um detalhe importante, o Delphi6 deve estar atualizado para o Pack 2.

O arquivo .ZIP contendo o MidasLib.DCU e MidasLib.RES pode ser encontrado no endereço http://codecentral.borland.com/codecentral/ccweb.exe/listing?id=17562.
Dúvida enviada por Adriano José Rodrigues, Santa Cruz do Rio Pardo/SP

Pergunta: Criei um projeto com um Form e adicionei uma Unit. No Form eu possuo um botão e ao seu evento OnClik uma determinada procedure. Na Unit, eu criei uma procedure idêntica a do evento OnClik do Botão, mas quando tento definir a minha procedure da Unit ao evento OnClick do botão do Form dá erro. Por quê?

Exemplo:

```
unit Unit2;

interface

uses dialogs, unit1;

procedure MinhaProcedure(Sender: TObject);

implementation

procedure MinhaProcedure(Sender: TObject);
Begin
  ShowMessage('Minha Procedure');
```

```
  Form1.Button2.OnClick:=Form1.Button1.OnClick;
  // Sem problemas!
  Form1.Button2.OnClick:=minhaprocedure;
  // Erro!
end;

end.
```

Resposta: As procedures que representam um evento são consideradas métodos! Um método deve estar definido dentro de uma classe.
Observe que todos os métodos dos objetos dentro de uma unit, estão declarados dentro da classe do formulário:

```
type
  TForm1 = class(TForm)
    StatusBar1: TStatusBar;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

Tudo que você declarar fora da classe do formulário é considerada uma procedure regular e não um método.

Com isso, crie a sua procedure “Click” assim:

```
type
  TForm1 = class(TForm)
    StatusBar1: TStatusBar;
    Button1: TButton;
    procedure Meu_Clique(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

procedure TForm1.Meu_Clique(Sender: TObject);

begin
  ShowMessage('The Club');
end;
```

Com isso poderá atribuí-la aos seus objetos.

Dúvida enviada por Escape Soft BroadCast, Nova Friburgo/RJ

Pergunta: Teria alguma API que me possibilitasse verificar o estado do modem?

Resposta: Segue abaixo um exemplo de como verificar o Status do modem:

```
var
  CommPort : string;
  hCommFile : THandle;
  ModemStat : DWord;

begin
  CommPort := 'COM2';
  {Abrir por de comunicação}
  hCommFile := CreateFile(PChar(CommPort),
                          GENERIC_READ,
                          0,
                          nil,
                          OPEN_EXISTING,
                          FILE_ATTRIBUTE_NORMAL,
                          0);
  if hCommFile = INVALID_HANDLE_VALUE then

  begin
    ShowMessage('Não foi possível abrir a porta '+
        CommPort);
    exit;
  end;
  {Verificar status}
  if GetCommModemStatus(hCommFile, ModemStat) <>
        false then

  begin
    if ModemStat and MS_CTS_ON <> 0 then
      ShowMessage('The CTS (clear-to-send) is on.');
    if ModemStat and MS_DSR_ON <> 0 then
      ShowMessage('The DSR (data-set-ready) is on.');
    if ModemStat and MS_RING_ON <> 0then
      ShowMessage('The ring indicator is on.');
    if ModemStat and MS_RLSD_ON <> 0 then
      ShowMessage('The RLSD (receive-line-signal-
        detect) is on.');
  end;
  {fechar a porta}
  CloseHandle(hCommFile);
end;
```

Dúvida enviada por Venture Training & Informática, São Paulo/ SP